

IN THE  
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Coutant et al.

Confirmation No.: 1046

Application No.: 09/702,593

Examiner: Kendall, C.

Filing Date: 10/31/2000

Group Art Unit: 2192

Title: METHOD AND APPARATUS FOR SWITCHING BETWEEN MULTIPLE IMPLEMENTATIONS OF A ROUTINE

Mail Stop Appeal Brief-Patents  
Commissioner For Patents  
PO Box 1450  
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 09/01/2005.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

**(complete (a) or (b) as applicable)**

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

- |                  |           |
|------------------|-----------|
| ( ) one month    | \$120.00  |
| ( ) two months   | \$450.00  |
| ( ) three months | \$1020.00 |
| ( ) four months  | \$1590.00 |

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of \$500.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

( ) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:  
Commissioner for Patents, Alexandria, VA  
22313-1450. Date of Deposit: 11/01/2005  
OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number \_\_\_\_\_ on \_\_\_\_\_

Number of pages:

Typed Name: Kathleen Klinkhammer

Signature: Kathleen Klinkhammer

Respectfully submitted,

Coutant et al.

By LeRoy D. Maunu

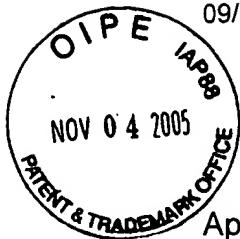
LeRoy D. Maunu

Attorney/Agent for Applicant(s)

Reg. No. **35,274**

Date: **11/01/2005**

Telephone No.: **(651) 686-6633**



09/702,593

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Coutant et al.	Examiner:	Kendall, C.
Serial No.:	09/702,593	Group Art Unit:	2192
Filed:	October 31, 2000	Docket No.:	10001275-1 (HPCO.008PA)
Title:	METHOD AND APPARATUS FOR SWITCHING BETWEEN MULTIPLE IMPLEMENTATIONS OF A ROUTINE		

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence and the papers, as described hereinabove, are being deposited in the United States Postal Service, as first class mail, in an envelope addressed to: Board of Patent Appeals and Interferences, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450, on November 1, 2005.

By: Kathleen Klinkhammer  
Kathleen Klinkhammer

Mail Stop Appeal Brief-Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

This is an Appeal Brief submitted pursuant to 37 C.F.R. § 41.37 for the above-referenced patent application.

**I. Real Party in Interest**

The real party in interest is Hewlett-Packard Company having a place of business at 1501 Page Mill Road, Palo Alto, CA. The above referenced patent application is assigned to Hewlett-Packard Company.

**II. Related Appeals and Interferences**

Appellant is unaware of any related appeals, interferences or judicial proceedings.

11/04/2005 NGUYEN1 00000077 082025 09702593

01 FC:1402 500.00 DA

### **III. Status of Claims**

Claims 1-16 are rejected and are presented for appeal. The appealed claims are in the attached Appendix of Appealed Claims.

### **IV. Status of Amendments**

No amendment has been filed subsequent to the final rejection.

### **V. Summary of Invention**

One embodiment of Appellant's invention is directed to a method for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system. A plurality of implementations of a routine are compiled into respective object code modules (FIG. 1, #108, #114; p. 3, l. 14 – p. 4, l. 21). The routine has an associated name and each implementation is adapted to a selected hardware configuration (FIG. 1, #104; p. 3, ll. 26-27; p. 3, l. 33 – p. 4, l. 16). The object code modules are associated with the name of the routine and respective sets of hardware characteristics (FIG. 2, #152, 154; p. 4, l. 30 – p. 5, l. 3). When the application program is loaded into memory of the computer system, a reference to the routine is resolved using the sets of hardware characteristics and a hardware configuration of the system (FIG. 3, #302, 306; p. 5, ll. 20-30).

In another embodiment, a computer-implemented method for switching includes establishing a set of hardware configuration characteristics that describe the computer system (FIG. 1, 104, 106; p. 3, l. 26 – p. 4, l. 4). A symbol table is established and includes one or more entries that include a name of a routine, a set of hardware characteristics, and an address referencing a routine in the library (FIG. 1, 106; p. 3, l. 26-29; FIG. 2, 154; p. 4, l. 30 – p. 5, l. 3). A name of a routine having multiple implementations is obtained when the library is loaded with the application program into memory of the computer system (FIG. 3, 304; p. 5, l. 25-28). The name of the routine and the set of hardware configuration characteristics that describe the computer system are matched to an entry in the symbol table (FIG. 3, 306; p. 5, l. 29-34), and an address in executable code is generated for references

to the routine having multiple implementations when the library is loaded with the application program; the address references an implementation in the library as identified in the matching step by the entry in the symbol table (FIG. 3, 306, 308; p. 5, l. 29-34).

An apparatus (p. 2, l. 7-15) for switching between multiple implementations of a routine is provided in another embodiment. The apparatus comprises: means for compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration (FIG. 1, #108, #114; p. 3, l. 14 – p. 4, l. 21); means for associating the object code modules with the name of the routine and respective sets of hardware characteristics (FIG. 2, #152, 154; p. 4, l. 30 – p. 5, l. 3); and means for resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system (FIG. 3, #302, 306; p. 5, ll. 20-30).

Another embodiment of the invention provides a computer-implemented symbol table for referencing a library of object code modules that implement a plurality of routines. The symbol table (FIG. 2, 154; p. 4, l. 23-25) includes a first set of one or more entries, each entry in the first set including a unique name (e.g., FIG. 2, "routine 1" – "routine  $n$ "; p. 4, l. 30-32) of a routine and a reference to an object code module in the library (p. 4, l. 30-32). The symbol table also includes a second set of one or more entries (e.g., FIG. 2, "routine  $n+1$ "; p. 4, l. 26-29). Each entry in the second set includes a shared name of a routine (e.g., FIG. 2, "routine  $n+1$ "), a set of hardware characteristics (e.g., FIG. 2, "routine  $n+1$ " and associated "set 1" hardware characteristics; p. 4, l. 30 – p. 5, l. 3), and a reference to an object code module in the library (FIG. 2, links to "routine  $n+1$  code" and to "routine ( $n+1$ )' code"; p. 4, l. 30 – p. 5, l. 3).

A computer program product (p. 2, l. 7-15) is provided in another embodiment. The computer program product is configured for causing a computer to perform the steps of: compiling a plurality of implementations of a routine into respective object code modules (FIG. 1, #108, #114; p. 3, l. 14 – p. 4, l. 21), the

routine having an associated name and each implementation adapted to a selected hardware configuration (FIG. 1, #104; p. 3, ll. 26-27; p. 3, l. 33 - p. 4, l. 16); associating the object code modules with the name of the routine and respective sets of hardware characteristics (FIG. 2, #152, 154; p. 4, l. 30 - p. 5, l. 3); and resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system (FIG. 3, #302, 306; p. 5, ll. 20-30).

## **VI. Grounds of Rejection**

- A. Claims 1-16 stand rejected under 35 U.S.C. §102(b) as being anticipated by "Davidson" (US patent 5,613,117 to Davidson et al.).

## **VII. Argument**

- A. **The rejection of claims 1-16 as being anticipated by Davidson should be reversed because the Examiner has not shown that Davidson teaches all the limitations of the claims.**

### Claims 1, 2, 7, 13, and 16

As to claim 1, the Examiner fails to show that Davidson teaches all the limitations. For example, the Examiner does not show that Davidson teaches resolving, when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system. Davidson's teachings at 27:23-32 are cited as corresponding to these limitations. However, there are no apparent elements of Davidson that correspond to these claim limitations as the cited text clearly demonstrates. The cited text reads:

The design of the compiler of FIG. 1 in general, and of the intermediate language and symbol table in particular, is intended to address a variety of architectures ranging from "Complex Instruction Set Computers" (CISC) such as VAX to "Reduced Instruction Set Computers" (RISC) such as PRISM, MIPS (a 32-bit RISC machine), or an advanced 64-bit RISC architecture. This design does assume that the architecture of target machine 25 has certain

basic features. First byte organization and addressability are assumed and Twos-complement binary arithmetic, with "Little-endian" bit ordering. "Reasonable" address representation is also assumed, i.e., that an address fits in a register.

There is no apparent use of multiple sets of hardware characteristics in combination with the hardware configuration of the host system in resolving a reference to a routine, as is clearly claimed. Nor does Davidson perform resolving when a program is loaded. Davidson has no apparent need for these limitations because object modules are created for only one selected target computer architecture (e.g., FIG. 1). Thus, Davidson apparently creates code for a chosen target computer architecture, and there is no apparent need for using the multiple sets of hardware characteristics to resolve a reference to a routine that has been compiled into a plurality of implementations.

The Examiner also fails to show that Davidson teaches associating the object code modules with the name of the routine and respective sets of hardware characteristics. Davidson's teachings at 6:30-50 and 9:8-15 are cited as corresponding to these limitations. However, there is no apparent association of a name with multiple sets of hardware characteristics. The cited sections of Davidson teach:

Third, the front end 20 does lexical, syntactic, and semantic analysis to translate the source text in file 21 to a language-independent internal representation used for the interface 22 between the front end 20 and the back end 12. Fourth, the front end 20 invokes the back end 12 to generate target system object code 23 from the information in the internal representation. Fifth, the front end 20 provides routines which the back end 12 calls via call path 24 to obtain language-specific information during back end processing. Not included in the compiler framework of FIG. 1 is a linker which links the object code modules or images 23 to form an executable image to run on the target machine 25.

The target machine 25 for which the back end 12 of the compiler creates code is a computer of some specific architecture, i.e., it has a register set of some specific number and data width, the logic executes a specific instruction set, specific addressing modes are available, etc. Examples are (1) the VAX architecture, as described in (2) a RISC type of architecture based upon the 32-bit RISC chip available from MIPS, Inc., ... (6:30-50)

...

To create an object module 23, the front end 20 translates the input stream or some subsequence thereof (which we can call a source module 21) into its internal representation for interface 22, which consists of a symbol table 30 for the module and an intermediate language graph 55 for each routine. The front end 20 then calls back end routines to initialize the object module 23, to allocate storage for the symbols in the symbol table 30 via storage allocation 28, to initialize that storage, to generate code for the routines via emitter 29, and to complete the object module 23 (9:8-18).

Davidson's compiler apparently has the capability to create an object code module for a selected target machine architecture. However, the fact that an object code module may be generated for a particular target machine is not suggestive of any association of a name of a routine with respective sets of hardware characteristics. Furthermore, there is no apparent need for Davidson to perform any of the claimed associating since Davidson creates the object code module for a target machine and not respective object code modules for selected hardware configurations.

Claims 2 and 7 depend from claim 1 and are not shown to be anticipated for at least the reasons set forth above.

Claim 13 is an apparatus claim that includes functional limitations similar to those of claim 1, and claim 16 is a computer program product claim having functional limitations similar to those of claim 1. Therefore, the Examiner has failed to show that Davidson teaches the limitations of claims 13 and 16 for at least the reasons set forth above.

The rejections of claims 1, 2, 7, and 16 should be reversed because the Examiner failed to show that Davidson teaches all the limitations of the claims.

#### Claims 3 and 4

Claim 3 depends from claim 2 and includes the further limitations: for the routine having a plurality of implementations, a plurality of entries are added to the symbol table and respective sets of hardware characteristics are associated with the plurality of entries. It is respectfully submitted that the Office Action fails to show any associating of entries in a symbol table with respective sets of hardware characteristics. The Office Action cites Davidson's symbol table and Davidson's assumed basic features of a hardware architecture (byte organization,

addressability, twos-complement arithmetic, and Little-endian). However, as explained above, Davidson's hardware assumptions obviate the need for any association as claimed. Furthermore, there is clearly no teaching or suggestion that Davidson performs any associating of entries in a symbol table with respective sets of hardware characteristics. Thus, claim 3 is not shown to be anticipated.

Claim 4 depends from claim 3 and is not shown to be anticipated for at least the reasons set forth above.

The rejections of claims 3 and 4 should be reversed because the Examiner failed to show that Davidson teaches all the limitations of the claims.

#### Claims 5, 6, 8

Claim 5 depends from claim 4, and claim 6 depends from claim 3. These claims include the further limitations of the resolving step including obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware. As explained above in regards to claim 1, Davidson is not shown to perform any resolving. Thus, the further limitations related to the resolving are also not shown to be taught by Davidson. Furthermore, the cited teaching of Davidson (6:43-47) simply indicates that the target machine for Davidson's backend is a computer with a specific architecture. There is no apparent suggestion of obtaining hardware configuration information from the specifically claimed sources. Therefore, claims 5, 6, and 8 are not shown to be anticipated by Davidson.

The rejections of claims 5, 6, and 8 should be reversed because the Examiner failed to show that Davidson teaches all the limitations of the claims.

#### Claim 9

The limitations of claim 9 are clearly not shown to be taught by Davidson. Claim 9 is an independent claim that includes limitations that further refine the limitations of claim 1 such as having a set of hardware characteristics in a symbol table. The cited section of Davidson contains no apparent teaching of hardware characteristics in a symbol table. The cited teachings of Davidson at 9:8-18 read:



To create an object module 23, the front end 20 translates the input stream or some subsequence thereof (which we can call a source module 21) into its internal representation for interface 22, which consists of a symbol table 30 for the module and an intermediate language graph 55 for each routine. The front end 20 then calls back end routines to initialize the object module 23, to allocate storage for the symbols in the symbol table 30 via storage allocation 28, to initialize that storage, to generate code for the routines via emitter 29, and to complete the object module 23.

Because this cited portion of Davidson contains no apparent elements that correspond to the claimed set of hardware characteristics being in the symbol table, Applicants in the Response of March 2, 2005, requested an explanation as to those elements of Davidson thought to correspond to the hardware characteristics being in the symbol table. However, the request went unanswered.

The cited teachings of Davidson at 9:20-30 are similarly lacking in any apparent teaching of, and lacking in any reasonable correspondence of elements to the further limitations of claim 9 including matching the name of the routine and the set of hardware configuration characteristics that describe the computer system to an entry in the symbol table. The Examiner has not provided an explanation of his understanding of the relevance of Davidson's elements as requested.

The rejection of claim 9 should be reversed because the Examiner failed to show that Davidson teaches all the limitations of the claim.

#### Claim 10

Claim 10 depends from claim 9 and includes limitations similar to those of claims 3 and 4. Therefore, the rejection of claim 10 should be reversed for at least the reasons set forth above for claims 3-4 and 9.

#### Claims 11 and 12

Claims 11 and 12 depend from claims 10 and 9, respectively, and include limitations similar to those of claims 5 and 6 as discussed above. Therefore, the rejection of claims 11 and 12 should be reversed for the reasons set forth above for claims 5-6 and 9.

Claims 14 and 15

Claim 14 sets forth a computer-implemented symbol table for referencing a library of object code modules that implement a plurality of routines. The limitations include a first set of one or more entries, each entry in the first set including a unique name of a routine and a reference to an object code module in the library; and a second set of one or more entries, each entry in the second set including a shared name of a routine, a set of hardware characteristics, and a reference to an object code module in the library.

The Examiner has not shown that Davidson teaches all the limitations of claim 14. As explained above in regards to claim 9, the Examiner has not shown that Davidson's symbol table includes hardware characteristics along with the routine names. Thus, the limitations of claim 14 are not shown to be taught by Davidson.

Claim 15 depends from claim 14, and is not shown to be anticipated for at least the reasons set forth above.


The rejection of claims 14 and 15 should be reversed because the Examiner failed to show that Davidson teaches all the limitations of the claims.

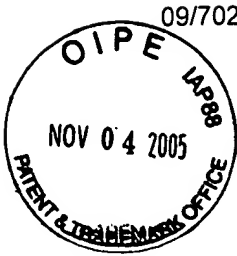
**VIII. Conclusion**

In view of the above, Appellant submits that the rejections are improper, the claimed invention is patentable, and that the rejections of claims 1-16 should be reversed. Appellant respectfully requests reversal of the rejections as applied to the appealed claims and allowance of the entire application.

Respectfully submitted,

CRAWFORD MAUNU PLLC  
1270 Northland Drive, Suite 390  
Saint Paul, MN 55120  
(651) 686-6633

By:   
Name: LeRoy D. Maunu  
Reg. No.: 35,274



09/702,593

**APPENDIX OF APPEALED CLAIMS  
FOR APPLICATION NO. 09/702,593**

1. A computer-implemented method for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system, comprising:
  - compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;
  - associating the object code modules with the name of the routine and respective sets of hardware characteristics; and
  - resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.
2. The method of claim 1, further comprising establishing a symbol table having a plurality of entries, each entry including a name of a routine and a reference to an object code module in the library.
3. The method of claim 2, further comprising, for the routine having a plurality of implementations, adding a plurality of entries to the symbol table and associating respective sets of hardware characteristics with the plurality of entries.
4. The method of claim 3, wherein the hardware characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system.

5. The method of claim 4, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.
6. The method of claim 3, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.
7. The method of claim 1, wherein the hardware characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, and instruction set characteristics.
8. The method of claim 1, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.
9. A computer-implemented method for switching between multiple implementations of a routine in a library of routines that are linked with an application program hosted by a computer system, comprising:
  - establishing a set of hardware configuration characteristics that describe the computer system;
  - establishing a symbol table, the symbol table having one or more entries that include a name of a routine, a set of hardware characteristics, and an address referencing a routine in the library;
  - obtaining a name of a routine having multiple implementations when the library is loaded with the application program into memory of the computer system;
  - matching the name of the routine and the set of hardware configuration characteristics that describe the computer system to an entry in the symbol table;
  - and

generating an address in executable code for references to the routine having multiple implementations when the library is loaded with the application program, the address referencing an implementation in the library as identified in the matching step by the entry in the symbol table.

10. The method of claim 9, wherein the hardware configuration characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, and instruction set characteristics.

11. The method of claim 10, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.

12. The method of claim 9, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.

13. An apparatus for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system, comprising:

means for compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;

means for associating the object code modules with the name of the routine and respective sets of hardware characteristics; and

means for resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.

14. A computer-implemented symbol table for referencing a library of object code modules that implement a plurality of routines, comprising:

a first set of one or more entries, each entry in the first set including a unique name of a routine and a reference to an object code module in the library; and

a second set of one or more entries, each entry in the second set including a shared name of a routine, a set of hardware characteristics, and a reference to an object code module in the library.

15. The symbol table of claim 14, wherein the hardware characteristics include at least one of clock speed of a processor, processor model, cache configuration, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system.

16. A computer program product configured for causing a computer to perform the steps of:

compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;

associating the object code modules with the name of the routine and respective sets of hardware characteristics; and

resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.

**APPENDIX OF EVIDENCE FOR  
APPLICATION NO. 09/702,593**

Appellant is unaware of any evidence submitted in this application pursuant to 37 C.F.R. §§ 1.130, 1.131, and 1.132.

**APPENDIX OF RELATED PROCEEDINGS FOR  
APPLICATION NO. 09/702,593**

Appellant is unaware of any related appeals, interferences or judicial proceedings.